Available online at www.sciencedirect.com

ScienceDirect

Journal of Computational and Applied Mathematics 218 (2008) 350–363

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

www.elsevier.com/locate/cam

Simple geometry facilitates iterative solution of a nonlinear equation via a special transformation to accelerate convergence to third order

M. Çetin Koçak*

Chemical Engineering Department, Engineering Faculty, Ankara University, 06100 Ankara, Turkey

Received 22 August 2006; received in revised form 24 February 2007

Abstract

Direct substitution $x_{k+1} = g(x_k)$ generally represents iterative techniques for locating a root z of a nonlinear equation $f(x)$. At the solution, $f(z) = 0$ and $g(z) = z$. Efforts continue worldwide both to improve old iterators and create new ones. This is a study of convergence acceleration by generating secondary solvers through the transformation $g_m(x) = (g(x) - m(x)x)/(1 - m(x))$ or, equivalently, through partial substitution $g_{m\mathrm{ps}}(x) = x + G(x)(g - x)$, $G(x) = 1/(1 - m(x))$. As a matter of fact, $g_m(x) \equiv g_{m\mathrm{ps}}(x)$ is the point of intersection of a linearised g with the $g = x$ line. Aitken's and Wegstein's accelerators are special cases of g_m. Simple geometry suggests that $m(x) = (g'(x) + g'(z))/2$ is a good approximation for the ideal slope of the linearised g. Indeed, this renders a third-order g_m. The pertinent asymptotic error constant has been determined. The theoretical background covers a critical review of several partial substitution variants of the well-known Newton's method, including third-order Halley's and Chebyshev's solvers. The new technique is illustrated using first-, second-, and third-order primaries. A flexible algorithm is added to facilitate applications to any solver. The transformed Newton's method is identical to Halley's. The use of $m(x) = (g'(x) + g'(z))/2$ thus obviates the requirement for the second derivative of $f(x)$. Comparison and combination with Halley's and Chebyshev's solvers are provided. Numerical results are from the square root and cube root examples.

MSC: 37M05; 65B99; 65H05

Keywords: Algebraic equations solvers; Iterative methods; Fixed-point iterations; Simulation; Convergence order; Direct substitution; Partial substitution; Halley's method; Newton's method; Acceleration of iteration

1. Introduction

Efficient solution techniques are required for nonlinear equations which partake of scientific, engineering, economy, and various other models. Simulation applications, especially dynamic ones, may need multiple runs each demanding thousands of times zeroes of algebraic equations coupled to differential equations. Often, either lack or intractability of an analytical solution directs one to harness an iterative technique for this task and face possibilities of slow convergence, non-convergence and divergence, in other words, inefficiency or failure.

Let z be a zero of an arbitrary nonlinear function $f(x)$, that is, let $f(z) = 0$. Further, appoint k to count the iterations. A fixed-point iteration method starts from one or more guessed x values, thereafter repeatedly uses direct substitution

* Tel.: +90 312 2126720.
E-mail address: kocak@eng.ankara.edu.tr.

0377-0427/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.cam.2007.02.036

$x_{k+1} = g(x_k)$ until it reaches a point where two consecutive x values coincide within a pre-specified tolerance, and reports this point as z.

Let ε_k denote the error in the kth iterate, that is, let $\varepsilon_k = x_k - z$. By definition, if $\lim_{k\to+\infty}(|\varepsilon_{k+1}|/|\varepsilon_k|^n) = c_{\text{as}}$, then n and c_{as} are the convergence order and the asymptotic error constant. "*Linear*" or "*first-order*" methods are those where $n = 1$, $g'(z) \neq 0$ and so $|\varepsilon_{k+1}|$ is proportional to $|\varepsilon_k|$ in the neighbourhood of z. For $n = 2$, the only requirement is $g'(z) = 0$ whereas for $n > 2$, this becomes $g'(z) = g''(z) = \cdots = g^{(n-1)}(z)$ and $g^{(n)}(z) \neq 0$.

Remark 1. nth order solvers are a subset of $(n-1)$th order solvers. Higher n may mean fewer but not necessarily cheaper iterations. Irrespective of n, there is always a risk of divergence if not started close enough to z. Iteration equations obtained from a rearrangement of $f(x) = 0$ are mostly linear. (See illustrations 1–3 presented later.)

Beyond this point in the text, when functions are written without an argument the latter is x. Newton's popular technique [1,3–5,13] is a computationally simple, one-point method without memory: $g_{\text{N}} = x - f/f'$. It is a piecewise linearisation of f since it extends the current tangent to intersect the x-axis and suggests this value as the next approximation to z. For simple roots this method is second order. Repetition of z denotes convergence from quadratic to *superlinear* or *geometrical* and that slows down the iteration process. If r is the multiplicity of z, then $g'(z) = (r-1)/r \neq 0$ and $g_{\text{N}r} = x - rf/f'$ restores second-order convergence [5].

Partial substitution is an attempt to improve convergence of a given solver by insertion of a variable gain G into the direct substitution formula so that it becomes $g_{\text{ps}} = x + G(g - x)$. Applied to Newton's scheme, partial substitution gives $g_{\text{Nps}} = x - Gf/f'$.

Remark 2. $g_{\text{N}r} = x - rf/f'$ is a g_{Nps} with a *fixed* $G = r$.

Chebychev's ($g_{\text{C}} = x - f/f' - f^2 f''/(2f'^3)$) and Halley's ($g_{\text{H}} = x - ff'/(f'^2 - 0.5ff'')$ methods are two well-known solvers that are of third order for simple roots. They are, like so many others, g_{Nps} variants (see Section 2). Needless to say, both g_{C} and g_{H} are more involved and demanding than g_{N} since they require f'' in addition to f and f'.

Efforts continue worldwide both to boost existing iterative solvers and develop new ones. The new transformation of this research has been applied to accelerate linear solvers besides higher order ones like g_{N}, $g_{\text{N}r}$, g_{C}, and g_{H}. Now, g_{N} and its g_{Nps} variants (including $g_{\text{N}r}$, g_{C}, and g_{H}) belong to set of solvers in the form $g_u = x + fu$ where u is a weight function. With an obvious notation, $u_{\text{N}} = -1/f'$ and $u_{\text{Nps}} = -G/f'$.

2. Some solvers of the type $g_u = x + fu$

Direct differentiation of $g_u = x + fu$ with respect to x gives

$$g'_u = 1 + f'u + fu', \quad g''_u = f''u + 2f'u' + fu'', \quad g''' = f'''u + 3f''u' + 3f'u'' + fu'''.$$

As x goes to z the terms containing f disappear because $f(z) = 0$ and so these derivatives tend towards

$$g'_u(z) = 1 + f'(z)u(z), \quad g''_u(z) = f''(z)u(z) + 2f'(z)u'(z), \quad g'''_u(z) = f'''(z)u(z) + 3f''(z)u'(z) + 3f'(z)u''(z).$$

Equating $g'(z)$, $g''(z)$, and $g'''(z)$ to zero and rearranging, one obtains the first three rungs up the ladder of convergence order for g_u methods:

$$g'_u(z) = 0 \Rightarrow \quad u(z) = -1/f'(z), \tag{1a}$$

$$g''_u(z) = 0 \Rightarrow \quad u'(z) = -0.5f''(z)u(z)/f'(z) = 0.5f''(z)/f'^2(z), \tag{1b}$$

$$g'''_u(z) = 0 \Rightarrow \quad u''(z) = -(f'''(z)u(z) + 3f''(z)u'(z))/(3f'(z)) = \frac{f'''(z)}{3f'^2(z)} - \frac{f''^2(z)}{2f'^3(z)}. \tag{1c}$$

Remark 3. The implicit condition in (1) that $f'(z) \neq 0$ means z must be simple root ($r = 1$).

Violation of (1a) is enough to make g_u a first-order process. If u satisfies only (1a), then g_u is a second-order process. If u satisfies (1a) and (1b) but fails (1c), then $n = 3$. If u fulfils (1a), (1b), and (1c), then n is (at least) 4. It is relatively

easy to find a function u that tends towards $-1/f'(z)$ in the limit, thereby satisfying (1a) and rendering a second-order method, but it is a formidable task to fulfil (1b) in addition and rise to third order. It is even more difficult to satisfy (1c) as an extra step and climb to fourth order. Consider g_{N} again for which

$$u_{\mathrm{N}}(z) = -\frac{1}{f'(z)}, \quad u'_{\mathrm{N}}(z) = \frac{f''(z)}{f'^2(z)}, \quad u''_{\mathrm{N}}(z) = \frac{f'''(z)}{f'^2(z)} - \frac{2f''^2(z)}{f'^3(z)}.$$

Since $u_{\mathrm{N}}(z)$ exactly satisfies (1a) but both $u'_{\mathrm{N}}(z)$ and $u''_{\mathrm{N}}(z)$ fail to meet their respective conditions, g_{N} is utmost second order unless $f''(z) = 0$ in which case $n = 3$. It is noteworthy, however, that $u'_{\mathrm{N}}(z)$ and $u''_{\mathrm{N}}(z)$ fail to satisfy (1b) and (1c) because of coefficient discrepancies. This indicates that g_{N} has a good prospect of improvement and acceleration as exemplified below.

Assume that z is a simple zero. It is easy to show that

$$g'_{\mathrm{Nps}}(z) = 1 - G(z), \quad g''_{\mathrm{Nps}}(z) = \frac{f''(z)}{f'(z)}G(z) - 2G'(z).$$

It follows that g_{Nps} is (at least) third order if and only if

$$G(z) = 1, \quad G'(z) = 0.5f''(z)/f'(z).$$

Now, let L be the so-called "*logarithmic degree of convexity*", that is $L = f''f/f'^2$. Differentiation gives

$$L' = \frac{f''}{f'} - f\left(\frac{2f''^2}{f'^3} - \frac{f'''}{f'^2}\right), \quad L'(z) = \frac{f''(z)}{f'(z)}.$$

If a g_{Nps} with $G = H(L)$ is to reach third order, then H must be so chosen that $H(0) = 1$ and $H'(0) = 0.5$ because $G(z) = H(0)$ and $G'(z) = H'(z)L'(z)$. In translating Halley's own derivation into modern mathematics, this kind of reasoning lead Gander [7] to put many well-known third-order methods in a g_{Nps} form, namely,

$$g_{\mathrm{G}} = x - Gf/f', \quad G = H(L), \quad H(0) = 1, \quad H'(0) = 0.5.$$

The first one of these was Halley's solver[1]

$$g_{\mathrm{H}} = x - G\frac{f}{f'} = x - H(L)\frac{f}{f'}, \quad H(L) = \left(1 - \frac{1}{2}L\right)^{-1} = 1 + \frac{1}{2}L + \frac{1}{4}L^2 + \cdots.$$

The second was Euler's method[2]

$$g_{\mathrm{E}} = x - G\frac{f}{f'} = x - H(L)\frac{f}{f'}, \quad H(L) = 2(1 + \sqrt{1 - 2L})^{-1} = 1 + \frac{1}{2}L + \frac{1}{2}L^2 + \cdots.$$

There was a Hansen–Patrick family

$$g_{\mathrm{H\text{–}P}} = x - G\frac{f}{f'} = x - H(L)\frac{f}{f'}, \quad H(L) = (b+1)(b + \sqrt{1 - (b+1)L})^{-1} = 1 + \frac{1}{2}L + \frac{b+3}{8}L^2 + \cdots.$$

Then came Ostrowski's square root iteration

$$g_{\mathrm{O}} = x - G\frac{f}{f'} = x - H(L)\frac{f}{f'}, \quad H(L) = (1 - L)^{-0.5} = 1 + \frac{1}{2}L + \frac{3}{8}L^2 + \cdots,$$

and quadratic inverse interpolation

$$g_{\mathrm{qii}} = x - G\frac{f}{f'} = x - H(L)\frac{f}{f'}, \quad H(L) = 1 + \frac{1}{2}L.$$

---

[1] Deiters [2] called it "Kepler's method". According to Traub [13] "Halley's is *one the most frequently rediscovered methods in the literature*".

[2] Pakdemirli and Boyacı [11] reported this as "Householder's method".

Regarding them as quadratic curves that were tangent to f at the current iterate, Sharma [12] collected in a one-parameter family several solvers including g_{N}, g_{E}, g_{H}, super-Halley and Chebychev's methods. The latter two were, respectively,

$$g_{\text{s-H}} = x - G\frac{f}{f'} = x - H(L)\frac{f}{f'}, \quad H(L) = \left(1 - \frac{1}{2}L\right)(1-L)^{-1} = 1 + \frac{1}{2}L + \frac{1}{2}L^2 + \cdots,$$

$$g_{\mathrm{C}} = x - G\frac{f}{f'} = x - H(L)\frac{f}{f'}, \quad H(L) = 1 + \frac{1}{2}L.$$

The family was third order—with the single exception of g_{N}.

Remark 4. Notice that $g_{\mathrm{E}} \equiv g_{\text{s-H}}$ and $g_{\mathrm{C}} \equiv g_{\mathrm{qii}}$.

Koçak [8] also created a third-order class (g_{K}) in g_u form. Currently, this has an *invert-and-average* (g_{Kia}), an *average-and-invert* (g_{Kai}), and an *invert-and-exponentiate* (g_{Ke}) subclass. The g_{Kia} branch harnesses the average of two derivative reciprocals in u. Its general appearance is

$$g_{\mathrm{Kia}} = x + fu, \quad a = \frac{1}{f'_{\mathrm{s}}}, \quad u = -0.5\left(\frac{1}{f'} + a\right)$$

differences arising only from the choice of the secondary derivative f'_{s}. The recommended member g_{KiaN} relies on the highly popular g_{N}:

$$g_{\mathrm{KiaN}} = x + fu, \quad x_{\mathrm{X}} = g_{\mathrm{N}}, \quad a = \frac{1}{f'(x_{\mathrm{X}})}, \quad u = -0.5\left(\frac{1}{f'} + a\right).$$

The g_{Kai} subclass employs the reciprocal of the average of two derivatives in u as follows:

$$g_{\mathrm{Kai}} = x + fu, \quad u = -1/(0.5(f' + d)), \quad d = f'_{\mathrm{s}}.$$

Difference between subclass members arises from the choice of the secondary first derivative f'_{s}. On the other hand, the u of g_{KE} incorporates as a factor an exponentiated ratio of some f' and f'' values:

$$c = 0.5 f''_2/f'^2_1, \quad E = \mathrm{e}^{cf}, \quad u = -E/f', \quad g_{\mathrm{KE}} = x + fu.$$

The subclass member g_{Ea} uses analytical f' and f'' in c, viz.

$$c = 0.5 f''/f'^2, \quad E = \mathrm{e}^{cf}, \quad u = -E/f', \quad \text{and} \quad g_{\mathrm{KEa}} = x + fu.$$

Remark 5. Obviously, g_{KE} subclass is a g_{Nps} with $G = E$. Furthermore, g_{KEa} is akin and complementary to g_{H}, $g_{\mathrm{C}} \equiv g_{\mathrm{qii}}$, $g_{\mathrm{E}} \equiv g_{\text{s-H}}$, $g_{\text{H–P}}$, g_{O}, and other methods in Gander's g_{Nps} form $g_{\mathrm{G}} = x - Gf/f'$, $G = H(L)$, $L = f''f/f'^2$ since

$$cf = (0.5 f''/f')f = 0.5L, \quad E = \mathrm{e}^{cf} = \mathrm{e}^{0.5L} = 1 + \frac{1}{2}L + \frac{1}{2!2^2}L^2 + \frac{1}{3!2^3}L^3 \cdots.$$

Frontini and Sormani [6] derived an interesting family of *modified Newton's methods* by combining g_{N}, a general interpolating quadrature formula $Q_j(f)$, and the indefinite integral as follows. Let $x_i = x_k + d_i(g_{\mathrm{N}}(x_k) - x_k)$, $d_i \in [0, 1]$, $i = 1(1)j$ and $\sum_{i=1}^{j} A_i = 1$. From

$$f(x) = f(x_k) + \int_{x_k}^{x} f'(t)\,\mathrm{d}t = f(x_k) + Q_j(f) = 0 \;\Rightarrow -f(x_k) = Q_j(f) = (x - x_k)\sum_{i=1}^{j} A_i f'(x_i).$$

A final rearrangement gives

$$g_{\text{F–S}} = x = x_k - \frac{f(x_k)}{\sum_{i=1}^{j} A_i f'(x_i)}.$$

This is a family of type g_u where $u = -1/\sum_{i=1}^{j} A_i f'(x_i)$. Its n was shown to be 3 for simple and 1 for repeated roots. Two suggestions were made to raise n to 2 when the multiplicity r was known.

Iterative process of type g_u can be accelerated by a scheme due to Nedzhibov [10] as follows:

$$g_{\mathrm{Nd}} = x + fu\frac{f^{1/r}}{f^{1/r} - f^{1/r}(g)}.$$

This acceleration was claimed to raise the convergence order by at least 1. So, $g_{\mathrm{N}r}$ pushed in this fashion or

$$g_{\mathrm{NdN}r} = x - r\frac{f}{f'}\frac{f^{1/r}}{f^{1/r} - f^{1/r}(g_{\mathrm{N}r})}$$

is a third-order g_{Nps}.

Two well-known multi-point accelerators, namely Aitken's [1] and Wegstein's [4], are based on a piecewise linearisation of g. Both of them approximate g by a straight line with a slope m through the point $K(x_k, g_k)$, extend this line to intersect the $g = x$ line at $M(g_m, g_m)$, and then suggest that g_m be tried in the next iteration as the new guess for z. Their linearised g equation is

$$g_{\mathrm{lin}} - g_k = m(x - x_k), \quad m \neq 1,$$

and inserting $g_{\mathrm{lin}} = x = g_m$ into this yields after a rearrangement

$$g_m = \frac{g_k - mx_k}{1 - m}. \tag{2a}$$

Therefore, their prediction g_m is the point of intersection of g_{lin} with the $g = x$ line.

The difference between the two methods comes from the choice of m as explained now. Wegstein's KM line goes through a previous iterate $I(x_i, g_i)$, that is,

$$m = \frac{g_k - g_i}{x_k - x_i}, \quad g_{\mathrm{W}} = g_m = \frac{g_k x_i - g_i x_k}{x_i - x_k - (g_i - g_k)}.$$

The first g_{W} is calculable at the end of the second iteration and is updated at *each* iteration afterwards.

Aitken's KM line also goes through a previous iterate $I(x_i, g_i)$ but now $i = k - 1$ and $x_k = g_{k-1}$:

$$m = \frac{g_k - g_{k-1}}{x_k - x_{k-1}}, \quad g_{\mathrm{A}} = g_m = \frac{g_k x_{k-1} - g_{k-1}^2}{x_{k-1} - x_k - (g_{k-1} - g_k)} = g_k - \frac{(g_k - g_{k-1})^2}{x_{k-1} - x_k - (g_{k-1} - g_k)}.$$

The first g_{A} is calculable at the end of the second iteration and is updated *every other* iteration afterwards.

3. This work

Any solver g can generate countless secondary iterators g_m via the transformation

$$g_m = (g - mx)/(1 - m), \quad m \neq 1. \tag{2b}$$

This equation is a rearrangement of the basic direct substitution form $x = g$ after subtracting mx from both sides. Members of this g_m family display different convergence behaviour depending on the variable parameter m but they all have the same fixed-points as g, that is $g_m(z) = g(z) = z$.

Comparison of (2a) with (2b) reveals that g_{A} and g_{W} are a subset of secondary solvers g_m. Perseverance yielded *a third-order*, *one-point* accelerator from this transformation. Returning to the xg-plane, the KM line has the slope

$$m = (g_m - g_k)/(g_m - x_k). \tag{3a}$$

For the point $M(g_m, g_m)$ to coincide with the target $Z(z, z)$, the obvious "*ideal*" action is to employ

$$m_{\mathrm{id}} = (z - g_k)/(z - x_k), \tag{3b}$$

and hence

$$g_{m,\mathrm{id}} = (g_k - m_{\mathrm{id}} x_k)/(1 - m_{\mathrm{id}}) = z. \tag{2c}$$

Remark 6. This horizontal line is the ideal for all g, that is, $g_{m,\mathrm{id}} \equiv g_{\mathrm{id}} = z$.

Since $(z - g_k)/(z - x_k) = \varepsilon_{k+1}/\varepsilon_k$, by definition,

$$m_{\mathrm{id}} = (z - g_k)/(z - x_k) = \varepsilon_{k+1}/\varepsilon_k. \tag{3c}$$

$g_{m,\mathrm{id}} \equiv g_{\mathrm{id}} = z$ has the highest possible convergence order since all its derivatives are zero. It annihilates the error at once. Indeed, a single iteration may be performed optionally only to verify that the new process has hit z. Albeit, m_{id} has to be approximated repeatedly until z accrues. A third-order g_m is proposed here as an accelerator based on a special m_{id} approximation. It is then proved, illustrated, and compared with other solvers.

Remark 7. $g_{m,\mathrm{id}} \equiv g_{\mathrm{id}} = z$ can be harnessed in post priori analysis, comparative studies and troubleshooting.

The following will be proved now:

(I) $m_{\mathrm{id}} = (z - g_k)/(z - x_k) = \varepsilon_{k+1}/\varepsilon_k = g'(z) + g''(z)\varepsilon_k/2! + g'''(z)\varepsilon_k^2/3! + g^{(\mathrm{iv})}(z)\varepsilon_k^3/4! + \cdots$,
(II) g_m is a partial substitution given by $g_{m\mathrm{ps}} = x + G(g - x)$, $G = 1/1 - m$, $m = 1 - 1/G$,
(III) $g_m'(z) = 0$ if $m(z) = g'(z)$,
(IV) $g_m''(z) = 0$ if $m'(z) = g''(z)/2$,
(V) $g_m'''(z) = 0$ if $m''(z) = g'''(z)/3$,
(VI) g_m is third order if $m = 0.5(g' + g'(z))$,
(VII) application of g_m with $m = 0.5(g' + g'(z))$ to g_{N} is identical with g_{H}.

Proof I. From Taylor's expansion of $g(x_k)$,

$$x_{k+1} = g(x_k) = g(z + \varepsilon_k) = g(z) + g'(z)\varepsilon_k + g''(z)\varepsilon_k^2/2! + g'''(z)\varepsilon_k^3/3! + \cdots. \tag{4}$$

Since $z = g(z)$, it follows that

$$\begin{aligned}\varepsilon_{k+1} &= x_{k+1} - z = x_{k+1} - g(z)\\ &= g'(z)\varepsilon_k + g''(z)\varepsilon_k^2/2! + g'''(z)\varepsilon_k^3/3! + \cdots.\end{aligned} \tag{5}$$

Remark 8. Suppose the first term dominates here so that $\varepsilon_{k+1} \cong g'(z)\varepsilon_k$. If $g'(z)$ is negative, then g is oscillatory. Partial substitution is useful here.

Rearranging (5),

$$\varepsilon_{k+1}/\varepsilon_k = g'(z) + g''(z)\varepsilon_k/2! + g'''(z)\varepsilon_k^2/3! + g^{(\mathrm{iv})}(z)\varepsilon_k^3/4! + \cdots.$$

Combining this with (3c) ends Proof I:

$$m_{\mathrm{id}} = \varepsilon_{k+1}/\varepsilon_k = g'(z) + g''(z)\varepsilon_k/2! + g'''(z)\varepsilon_k^2/3! + g^{(\mathrm{iv})}(z)\varepsilon_k^3/4! + \cdots. \qquad \square \tag{6}$$

Proof II. By direct substitution of the specified G,

$$g_{m\mathrm{ps}} = x + \frac{1}{1-m}(g - x) = \frac{x - mx + g - x}{1 - m} = \frac{g - mx}{1 - m} = g_m. \tag{7}$$

Thus, g_m and $g_{m\mathrm{ps}}$ can be harnessed interchangeably. $\square$

Proof III. Differentiation of g_m with respect to x yields

$$g'_m = \frac{(g' - (m'x + m))(1 - m) - (-m')(g - mx)}{(1 - m)^2} = \frac{m^2 - (1 + g')m + g' + m'(g - x)}{(1 - m)^2}$$
$$= \frac{(m - 1)(m - g') + m'(g - x)}{(1 - m)^2} \tag{8a}$$

and so

$$g'_m(z) = \frac{(m(z) - 1)(m(z) - g')}{(1 - m(z))^2} \tag{8b}$$

because $g(z) = z$. Remember that $m \neq 1$. Thus, $g'_m(z) = 0$ if $m(z) = g'(z)$. The proof is complete. □

Proof IV. Differentiation of g'_m with respect to x renders

$$g''_m = \frac{[(g'' - 2m')(1 - m) + m''(g - x)][1 - m]^2 - [2(1 - m)m'][(m - 1)(m - g') + m'(g - x)]}{(1 - m)^4} \tag{9a}$$

which in turn yields

$$g''_m(z) = \frac{(g''(z) - 2m'(z))}{1 - g'(z)}. \tag{9b}$$

Obviously, $g''_m(z) = 0$ if $m' = 0.5g''(z)$. □

Proof V. Equivalence of g_m and $g_{m\mathrm{ps}}$ will be utilised to get $m''(z)$ that makes $g'''_m(z) = 0$. From $g_{m\mathrm{ps}}$

$$g'_{m\mathrm{ps}} = 1 + G'(g - x) + G(g' - 1),$$
$$g''_{m\mathrm{ps}} = G''(g - x) + 2G'(g' - 1) + Gg'',$$
$$g'''_{m\mathrm{ps}} = G'''(g - x) + 3G''(g' - 1) + 3G'g'' + Gg'''.$$

Since $g(z) = z$, as x goes to z these derivatives tend towards

$$g'_{m\mathrm{ps}}(z) = 1 + G(z)(g'(z) - 1),$$
$$g''_{m\mathrm{ps}}(z) = 2G'(z)(g'(z) - 1) + G(z)g''(z),$$
$$g'''_{m\mathrm{ps}}(z) = 3G''(z)(g'(z) - 1) + 3G'(z)g''(z) + G(z)g'''(z).$$

Equating them to zero and rearranging, one obtains

$$g'_{m\mathrm{ps}}(z) = 0 \Rightarrow \quad G(z) = \frac{1}{1 - g'(z)}, \tag{10a}$$

$$g''_{m\mathrm{ps}}(z) = 0 \Rightarrow \quad G'(z) = \frac{G(z)g''(z)}{2(1 - g'(z))} = \frac{g''(z)}{2(1 - g'(z))^2}, \tag{10b}$$

$$g'''_{m\mathrm{ps}}(z) = 0 \Rightarrow \quad G''(z) = \frac{G(z)g'''(z)}{3(1 - g'(z))} + \frac{G'(z)g''(z)}{1 - g'(z)} = \frac{g'''(z)}{3(1 - g'(z))^2} + \frac{g''^2}{2(1 - g'(z))^3}. \tag{10c}$$

These G, G', and G''values must also make zero the equivalent g_m derivatives. On the other hand, differentiating $m = 1 - 1/G$ twice one obtains

$$m' = \frac{G'}{G^2}, \quad m'' = \frac{G''}{G^2} - \frac{2G'^2}{G^3}.$$

Using the results of (10),

$$m'(z) = \frac{G'(z)}{G^2(z)} = 0.5g''(z), \tag{11a}$$

$$\begin{aligned} m''(z) &= \frac{G''(z)}{G^2(z)} - \frac{2G'^2(z)}{G^3(z)} \\ &= \left(\frac{g'''(z)}{3(1-g'(z))^2} + \frac{g''^2(z)}{2(1-g'(z))^3}\right)(1-g'(z))^2 - 2\left(\frac{g''(z)}{2(1-g'(z))^2}\right)^2(1-g'(z))^3 \\ &= \frac{g'''(z)}{3} + \frac{g''^2(z)}{2(1-g'(z))} - \frac{g''^2(z)}{2(1-g'(z))}. \end{aligned} \tag{11b}$$

Hence, $m''(z) = g'''(z)/3$ is the condition for $g_m'''(z) = 0$. This ends the proof. □

Remark 9. It follows from (6) that $m_{\mathrm{id}}(z) = g'(z), m_{\mathrm{id}}'(z) = g''(z)/2, m_{\mathrm{id}}''(z) = g'''(z)/3, \ldots, m_{\mathrm{id}}^{(i-1)}(z) = g^{(i)}(z)/i, \ldots$. Comparison with $m(z) = g'(z), m'(z) = g''(z)/2, m''(z) = g'''(z)/3$, obtained above intuitively leads to the generalisation that the condition for m to render a g_m of order n, $n \geqslant 2$, is

$$g_m^{(i)}(z) = 0, \quad m^{(i-1)}(z) = g^{(i)}(z)/i, \quad i = 1, 2, \ldots, n-1.$$

Case 1: Convergence of g_m is linear. Here, $g_m'(z) \neq 0$ because $m(z) \neq g'(z)$. If $m(z)$ is between 1 and $g'(z)$, then according to (8b) $g_m'(z)$ is negative and so the first-order g_m is oscillatory—a situation to be avoided.

Case 2: Convergence of g_m is quadratic. The condition that $g_m'(z) = 0$ is satisfied by any m such that $\lim_{x\to z} m = g'(z)$. Since this is a limit condition there is much room to choose m and pass the threshold to higher n. For instance, m can be equal to $\mathrm{e}^{fF} g_k'$, $\mathrm{e}^{fF} g'(z)$ or a linear combination of these two where F is some function of x. This is easy to see considering that when x tends towards z, $f = 0$, the exponential factor is 1, and $g_k' = g'(z)$. The exponential factor is left out now for simplicity.

Remark 10. Employment of $m = g_k'$ in (2b) is equivalent to applying g_{N} to solve a secondary function given by $f_{\mathrm{s}} = x - g$ since (omitting k)

$$g_{\mathrm{N}} = x - \frac{f_{\mathrm{s}}}{f'} = x - \frac{x-g}{1-g'} = \frac{g - g'x}{1-g'} = g_m, \quad m = g'.$$

Case 3: Convergence of g_m is third order. $g_m'(z) = g_m''(z) = 0$ is the condition that m should fulfil. As proved above, this implies $\lim_{x\to z} m = g'(z)$, $\lim_{x\to z} m' = 0.5g''(z)$. Now, use of $m = g'(z)$ results in $m'(z) = 0$ and so $n < 3$ unless $g''(z) = 0$. Similarly, utilisation of $m = g_k'$ ends up with $m'(z) = g''(z)$ and so $n < 3$ unless $g''(z) = 0$. It will be shown shortly that a linear combination of $m = g_k'$ and $m = g'(z)$ passes the third-order test.

According to the mean value theorem for derivatives,

$$m_{\mathrm{id}} = (z - g_k)/(z - x_k) = (g(z) - g_k)/(z - x_k) = g'(x_i), \quad x_i \varepsilon (x_k, z).$$

Note that this does not necessarily mean $g'(x_i)\varepsilon(g_k', g'(z))$. Thus, consider a linear combination

$$m_\lambda = \lambda g_k' + (1-\lambda)g'(z)). \tag{12}$$

Taylor's expansion of g_k' in the neighbourhood of z is

$$g_k' = g'(z + \varepsilon_k) = g'(z) + g''(z)\varepsilon_k + g'''(z)\varepsilon_k^2/2! + g^{(\mathrm{iv})}(z)\varepsilon_k^3/3! + \cdots$$

and so

$$m_\lambda = g'(z) + \lambda g''(z)\varepsilon_k + \lambda g'''(z)\varepsilon_k^2/2! + \lambda g^{(\mathrm{iv})}(z)\varepsilon_k^3/3! \cdots.$$

Combining this with (6) derived above,

$$m_{\mathrm{id}} = \varepsilon_{k+1}/\varepsilon_k = m_\lambda + \left(\frac{1}{2!} - \lambda\right) g''(z)\varepsilon_k + \left(\frac{1}{3!} - \frac{\lambda}{2!}\right) g'''(z)\varepsilon_k^2 + \left(\frac{1}{4!} - \frac{\lambda}{3!}\right) g^{(\mathrm{iv})}(z)\varepsilon_k^3 + \cdots.$$

It is clear that $\lambda = \frac{1}{2}$ annihilates the second term yielding

$$m_{\mathrm{id}} = \varepsilon_{k+1}/\varepsilon_k = m_{1/2} + \left(\frac{1}{3!} - \frac{1}{2!2}\right) g'''(z)\varepsilon_k^2 + \left(\frac{1}{4!} - \frac{1}{3!2}\right) g^{(\mathrm{iv})}(z)\varepsilon_k^3 + \cdots. \tag{13}$$

If the third and higher derivatives of g are zero at z, then $m_{\mathrm{id}} = m_{1/2} = (g'_k + g'(z))/2$ exactly and so theoretically $g_m = z$.

Proof VI. With $m = (g' + g'(z))/2$,

$$\lim_{x\to z} m = g'(z), \quad \lim_{x\to z} m' = 0.5g''(z), \quad \lim_{x\to z} m'' = 0.5g'''(z) \tag{14}$$

and so

$$g'_m(z) = g''_m(z) = 0 \quad \text{but} \quad g'''_m(z) \neq 0$$

unless $g'''(z) = 0$. This proves that g_m is third order ($n = 3$).

The next step would be to establish the asymptotic error constant c_{as} which needs an estimate of $g'''_{m\mathrm{ps}}(z) \equiv g'''_m(z)$. An equation given in Proof V is

$$g'''_{m\mathrm{ps}}(z) = 3G''(z)(g'(z) - 1) + 3G'(z)g''(z) + G(z)g'''(z). \tag{15}$$

On the other hand,

$$G = 1/(1-m), \quad G' = m'/(1-m)^2, \quad G'' = (m''(1-m) + 2m'^2)/(1-m)^3,$$

and so using (14)

$$\begin{aligned} G(z) &= \frac{1}{1-g'(z)}, \\ G'(z) &= \frac{g''(z)}{2(1-g'(z))^2}, \\ G''(z) &= \frac{0.5g'''(z)(1-g'(z)) + 20.5^2 g''^2(z)}{(1-g'(z))^3} = \frac{g'''(z)(1-g'(z)) + g''^2(z)}{2(1-g'(z))^3}. \end{aligned} \tag{16}$$

Inserting (16) into (15)

$$\begin{aligned} g'''_{m\mathrm{ps}}(z) &= \frac{3}{2}\,\frac{g'''(z)(1-g'(z)) + g''^2(z)}{(1-g'(z))^3}(g'(z)-1) + 3\frac{g''(z)}{2(1-g'(z))^2}g''(z) + \frac{1}{1-g'(z)}g'''(z) \\ &= \frac{(1-1.5)g'''(z)}{1-g'(z)} - \frac{1.5g''^2(z)}{(1-g'(z))^2} + \frac{1.5g''^2(z)}{(1-g'(z))^2}, \\ \therefore\ g'''_{m\mathrm{ps}}(z) &= -\frac{0.5g'''(z)}{1-g'(z)}. \qquad \square \end{aligned}$$

A theorem adapted from [1] is as follows:

Theorem. *Assume that z is a root of $x = g(x)$, and that g is n times continuously differentiable for all x near z, for some $n \geqslant 2$. Furthermore, assume $g'(z) = \cdots = g^{(n-1)}(z) = 0$. Then if the initial guess x_1 is chosen sufficiently close to z, the iteration $x_{k+1} = g(x_k)$, $k \geqslant 1$ will have order of convergence n, and*

$$\lim_{k\to\infty} \frac{z - x_{k+1}}{(z - x_k)^n} = (-1)^{n-1} \frac{g^{(n)}(z)}{n!}.$$

Therefore, applying this theorem to g_m with $m = (g' + g'(z))/2$,

$$\lim_{k\to\infty} \frac{z - x_{k+1}}{(z - x_k)^3} = \frac{g'''_{m\mathrm{ps}}(z)}{3!} = -\frac{0.5}{3!} \frac{g'''(z)}{1 - g'(z)}.$$

Then, by definition,

$$c_{\mathrm{as}} = \frac{0.5}{3!} \left| \frac{g'''(z)}{1 - g'(z)} \right|.$$

Proof VII. Remember that third-order Halley's method is a partial substitution variant of second-order Newton's technique.

$$g_{\mathrm{H}} = x - G_{\mathrm{H}} \frac{f}{f'} = x - H(L) \frac{f}{f'}, \quad H(L) = \left(1 - \frac{1}{2}L\right)^{-1} = 1 + \frac{1}{2}L + \frac{1}{4}L^2 + \cdots.$$

On the other hand, the $g_m \equiv g_{m\mathrm{ps}}$ application to g_{N} is a g_{Nps}:

$$g_m = g_{m\mathrm{ps}} = x - G_m \frac{f}{f'}, \quad G_m = \frac{1}{1 - m}, \quad m = (g'_{\mathrm{N}} + g'_{\mathrm{N}}(z))/2.$$

Since $g'_{\mathrm{N}} = 1 - (f'^2 - ff'')/f'^2 = ff''/f'^2 = L$ and $g'_{\mathrm{N}}(z) = 0$,

$$m = (g'_{\mathrm{N}} + g'_{\mathrm{N}}(z))/2 = \frac{1}{2}L, \quad G_m = \frac{1}{1 - m} = \left(1 - \frac{1}{2}L\right)^{-1} = H(L) \equiv G_{\mathrm{H}}$$

and so the proof is complete. □

Past this point, the new, third-order, one-point accelerator is marked with a star superscript:

$$m^* = m_{1/2} = (g'_k + g'(z))/2, \quad g^*_m = (g - m_{1/2}x)/(1 - m_{1/2})$$

or, equivalently,

$$m^* = m_{1/2} = (g'_k + g'(z))/2, \quad G_m = 1/(1 - m^*), \quad g^*_m = x + G_m(g - x).$$

Its algorithm comprises five distinct steps:

(a) Obtainment of g' and $g'(z)$,
(b) Calculation of $m_{1/2} = (g'_k + g'(z))/2$,
(c) Insertion of $m_{1/2}$ into $g^*_m = (g - m_{1/2}x)/(1 - m_{1/2})$,
(d) Simplification of g^*_m, and finally,
(e) Harnessing this improved suggestion g^*_m in place of the original g.

Cases in order of increasing complexity will illustrate the first four steps before a more practical extension to high order solvers. Subsequent numerical examples will cover the whole process.

Illustration 1: Consider the *square root* problem: $f = x^2 - N$, $N > 0$. The first-order solver $g = N/x$ *fails* because it keeps swinging from x_1 to some x_2 and back irrespective of the first guess x_1 unless it is accidentally z. Following the formal steps, the new scheme generates g_m^* from g:

$$g = N/x, \quad g' = -N/x^2, \quad g'(z) = -1,$$

$$m_{1/2} = (g'_k + g'(z))/2 = -\left(\frac{N}{x^2} + 1\right)\Big/ 2 = -\frac{N + x^2}{2x^2},$$

$$g_m^* = (g - m_{1/2}x)/(1 - m_{1/2}) = \frac{(N/x) + ((N + x^2)/2x^2)x}{1 + (N + x^2)/2x^2} = \frac{(3N + x^2)/x}{(3x^2 + N)/x^2} = \frac{(3N + x^2)x}{3x^2 + N}.$$

This g_m^* is coincident with application of g_{H} to f. The new scheme has made a flyer out of a non-convergent, first-order method. In contrast, the outcome of using $m = g'(z) = -1$ in (2b) is $g_m = (x + N/x)/2$. The latter is the usual computational formula for finding square roots. It is equivalent to applying g_{N} to f and is only second order. Therefore, it is strongly recommended here that it be replaced by g_m^*.

Illustration 2: In the *cube root problem*, $f = x^3 - N$, $N > 0$. The first-order solver $g = N/x^2$ fails because it has oscillatory divergence. Proceeding as before

$$g = N/x^2, \quad g' = -2N/x^3, \quad g'(z) = -2,$$

$$m_{1/2} = (g'_k + g'(z))/2 = -\left(\frac{2N}{x^3} + 2\right)\Big/ 2 = -\frac{N + x^3}{x^3},$$

$$g_m^* = (g - m_{1/2}x)/(1 - m_{1/2}) = \frac{N/x^2 + ((N + x^3)/(x^2)x)}{1 + ((N + x^3)/x^3)} = \frac{(2N + x^3)/x^2}{(2x^3 + N)/x^3} = \frac{(2N + x^3)x}{2x^3 + N}.$$

Again, this g_m^* is coincident with application of g_{H} to f. In contrast, using $m = g'(z) = -2$ in (2b) one gets $g_m = (N + 2x^3)/(3x^2)$ which is equivalent to applying g_{N} to f and is only second order.

Illustration 3: In the more general *pth root problem*, $f = x^p - N$, $N > 0$. Suppose the solver is $g = N/x^{p-1}$. Obtainment of g_m^* is summarised as follows:

$$g = N/x^{p-1}, \quad g' = -(p-1)N/x^p, \quad g'(z) = -(p-1),$$

$$m_{1/2} = (g'_k + g'(z))/2 = -\frac{(N + x^p)(p-1)}{2x^p},$$

$$g_m^* = (g - m_{1/2}x)/(1 - m_{1/2}) = \frac{((p+1)N + (p-1)x^p)x}{(p+1)x^p + (p-1)N}.$$

Once more, this g_m^* is coincident with application of g_{H} to f.

Illustration 4: g, g'_k and $g'(z)$ are the only information the accelerator needs to operate. It can also be applied to solvers whose convergence is higher than first order. Remember that $g'(z) = 0$ in these cases. When boosted by the new technique Newton's simple root formula g_{N} becomes Halley's formula g_{H}:

$$g_{\mathrm{N}} = x - f/f', \quad g'_{\mathrm{N}} = f''f/f'^2, \quad g'_{\mathrm{N}}(z) = 0,$$

$$m_{1/2} = (g'_{\mathrm{N}} + g'_{\mathrm{N}}(z))/2 = 0.5g'_{\mathrm{N}} = 0.5f''f/f'^2,$$

$$g_{m\mathrm{N}}^* = (g_{\mathrm{N}} - m_{1/2}x)/(1 - m_{1/2}) = x - 2ff'/(2f'^2 - f''f) = x - ff'/(f'^2 - 0.5f''f) \equiv g_{\mathrm{H}}.$$

See Proof VII. The practical significance of this is that estimate of g'_{N} obviates the need for f''.

Table 1
Algorithm Gmstar (dgz, epx, maxit, h, x1)

```
dx := 1e20, x := x1, k := 0
while abs(dx) > epx
  k := k + 1, gx := g(x)
  dg := (g(x + h) − gx)/h,  m := (dg + dgz)/2
  gm := (gx − mx)/(1 − m) or G := 1/(1 − m), gm := x + G*(gx − x)
  dx := gm − x
  if k >maxit, print error message, and stop
  x := gm
end while
```

Illustration 5: The new scheme can also force Newton's repeated root formula $g_{\mathrm{N}r}$. Here,

$$g_{\mathrm{N}r} = x - rf/f', \quad g'_{\mathrm{N}r} = 1 - r + rf''f/f'^2, \quad g'_{\mathrm{N}r}(z) = 0,$$

$$m_{1/2} = (g'_{\mathrm{N}r} + g'_{\mathrm{N}r}(z))/2 = 0.5g'_{\mathrm{N}r},$$

$$g^*_{m\mathrm{N}r} = (g_{\mathrm{N}r} - m_{1/2}x)/(1 - m_{1/2}) = x - \frac{2rff'}{(r+1)f'^2 - rf''f}.$$

Another way to get here is to use g_{N} as above but remembering that $g'(z) = (r-1)/r \neq 0$.

Applications to second-order g_{N} and $g_{\mathrm{N}r}$ have clearly shown that steps (a) and (d) may become highly cumbersome. Although more difficult to justify the effort, the scheme is capable of forcing third- or even higher-order solvers too. Symbolic languages may help here but this was not attempted in this study. As an alternative, an iteration loop may envelop the booster with some modification, omitting step (d) and calculating g'_k by numerical differentiation which of course means an extra g per step. Table 1 presents *Algorithm Gmstar* as a versatile version. Here, dg holds the numerical approximation to g'_k. For flexibility, it is assumed that $\mathrm{d}gz = g'(z)$ will be set externally before the entry and the primary solver g be supplied as a function. This has been implemented to push third-order methods g_{C} and g_{H}, respectively, to produce $g^*_{m\mathrm{C}}$ and $g^*_{m\mathrm{H}}$.

Remark 11. If preferred, g'_k may be approximated by $(g_k - g_i)/(x_k - x_i)$, $k \geqslant 2$ as in Wegstein's method.

4. Numerical results and discussion

The numerical examples and the three starting points are the same as used by Koçak [8]. Five methods are compared, namely g_{C}, g_{H}, g_{N}, $g^*_{m\mathrm{C}}$, and $g^*_{m\mathrm{H}}$. The latter is a double application of the new technique to g_{N}. Each method is allowed to iterate by direct substitution $x_{k+1} = g(x_k)$, until $|f| < 10^{-4}$ or $k > 10$. A more complicated example was given by Koçak [9] as the application of g_{N}, $g^*_{m\mathrm{N}} \equiv g_{\mathrm{H}}$, g_{KiaN}, and $g^*_{m\mathrm{KiaN}}$ to estimation of the boiling temperature (T) of an N-component mixture at a specified pressure (P). Remember that both g_{H} and g_{KiaN} are of third-order. The function to be solved was

$$f(T) = \sum_{i=1}^{N} y_i - 1, \quad y_i = x_i p_i^0(T)/P, \quad p_i^0(T) = \exp(A_{1i} + A_{2i}/(A_{3i} + T)), \ N = 3$$

with the derivative $f'(T) = -\sum_{i=1}^{N} y_i A_{2i}/(A_{3i} + T)^2$.

Table 2 depicts the numerical outputs for *the square root problem* where $f = x^2 - N$, $f' = 2x$, $f'' = 2$. The test choice $N = 100$ means that the roots are $z_1 = 10$ and $z_2 = -10$. Here, the focus is on the positive root. When started at $x = 1$, g_{C} jumps to the left of zero and then steadily marches towards the negative square root. Its pushed version $g^*_{m\mathrm{C}}$ manages to progress towards the positive root but with some oscillation. g_{N} steadily converges to the target after an initial spring to 50.5 but performs remarkably better when helped by the booster to give g_{H}. The double acceleration product $g^*_{m\mathrm{H}}$, on the other hand, goes to the other root. In the region where $x > z$, all boosts are improvements upon the original solver.

Table 2
Square root iterations: $x_{k+1} = g(x_k)$

k	g_C	g_H	g_N	g^*_{mC}	g^*_{mH}
Starting at $x = 1$					
1	−1174.62500	2.9223301	50.5000000	1.6413732	−3.9846673
2	−440.548224	7.1776428	26.240099	2.6196596	−12.8390710
3	−165.375812	9.9116812	15.0255301	3.8891687	−9.9876271
4	−62.4691656	9.9999983	10.8404347	4.7861714	−10.0000000
5	−24.6214018	10	10.0325785	3.6478361	−10.0000000
6	−12.1954086		10.0000529	4.7223154	
7	−10.0339723		10.0000000	3.8903646	
8	−10.0000002		10	4.7862937	
9	−10			3.6473191	
Starting at $x = 25$					
1	12.2950000	11.7088608	14.5000000	10.0436290	9.9678099
2	10.0381159	10.0097600	10.6982759	9.9999998	10.0000000
3	10.0000003	10.0000000	10.0227882	10.0000000	10.0000000
4	10	10	10.0000259		
5			10.0000000		
6			10		
Starting at $x = 50$					
1	20.2400000	18.4210526	26	14.0231983	12.8664352
2	11.1447762	10.5341401	14.9230769	9.9700993	9.9873401
3	10.0058842	10.0003520	10.8120539	10.00000000	10.0000000
4	10.0000000	10.0000000	10.0304952	10.00000000	10.0000000
5	10	10	10.0000464		
6			10.0000000		
7			10		

Table 3 shows the numerical outputs for the cube root problem where $f = x^3 - N = (x - \sqrt[3]{N})(x^2 + x\sqrt[3]{N} + \sqrt[3]{N^2})$, $f' = 3x^2$, $f'' = 6x$. The test choice $N = 1000$ means that the only real root is $z = 10$. When iteration starts at $x = 1$, g_C jumps to the left of zero and then steadily marches towards 0. g^*_{mC} progresses towards z with some oscillation. g_N steadily converges to the target after an initial spring to 50.5 but performs remarkably better when transformed by the booster to give g_H. g^*_{mH}, on the other hand, leaps to 167.5 and then approaches z steadily from the right. In the region where $x > z$, once again, all boosts are improvements upon the original solver.

It seems advisable, both in the square- and cube root cases, to keep iterations to the right of the root. Here, all of the five methods converge without oscillation and the improvement gained by acceleration is outstanding. $g^*_{mN} \equiv g_H$ is most recommendable.

5. Conclusions

The transformation

$$g_m(x) = (g(x) - m(x)x)/(1 - m(x)) \equiv x + G(x)(g(x) - x), \quad G(x) = 1/(1 - m(x))$$

is a superb acceleration facility to improve iterative solvers. The convergence of the secondary solver g_m is third order if $m(x) = (g'(x) + g'(z))/2$. First-, second-, third-order methods have been boosted here but the accelerator is general enough to push any iterative technique. Of course, benefits must be balanced against accompanying user labour and extra function counts. First-order methods show most improvement from this acceleration since it raises them to *third* order directly. When the novel technique is applied to push Newton's second-order method for simple roots, Halley's solver appears. The latter is recommended for use in finding pth root, square root in particular, instead of Newton's formula.

Table 3
Cube root iterations: $x_{k+1} = g(x_k)$

k	g_C	g_H	g_N	g^*_{mC}	g^*_{mH}
Starting at $x = 1$					
1	−110555	1.9970060	334	1.3999986	167.5005823
2	−61419	3.9470468	222.6696547	1.9565847	55.8929085
3	−34122	7.4257026	148.4531594	2.7228389	19.1590277
4	−18957	9.7953542	98.9838981	3.7473740	10.0219483
5	−10531	9.9999411	66.0232866	5.0116490	10.0000000
6	−5851	10.0000000	44.0919932	6.2430524	10.0000000
7	−3250	10	29.5661208	6.3835210	
8	−1806		20.0920676	6.1314176	
9	−1003		14.2204254	6.4988834	
10	−557		11.1286493	5.7740118	
Starting at $x = 20$					
1	12.4652778	11.7647059	14.1666667	10.4299336	10.0832721
2	10.1313575	10.0280996	11.1053441	9.9994817	9.9999981
3	10.0000363	10.0000001	10.1063677	10.0000000	10.0000000
4	10	10	10.0011156	10.0000000	10.0000000
5	10		10.0000000		
Starting at $x = 250$					
1	138.8977777	125.0119996	166.6720000	96.1737264	83.3599969
2	77.1942262	62.5539783	111.1266659	37.1240750	28.0267577
3	42.9788712	31.4682672	74.1114364	15.1614098	11.3047690
4	24.1771512	16.4795561	49.4683130	9.9675084	9.9950324
5	14.3687240	10.7239095	33.1150902	10.0000003	10.0000000
6	10.4920722	10.0022679	22.3806941	10.0000000	10.0000000
7	10.0017209	10.0000000	15.5859376		
8	10.0000000	10	11.7628107		
9	10.00000		10.2509832		
10	10		10.0060949		

References

[1] K.E. Atkinson, An Introduction to Numerical Analysis, Wiley, New York, 1978.
[2] U.K. Deiters, Calculation of densities from cubic equations of state, AIChE J. 48 (10) (2002) 882–886.
[3] L.V. Fausette, Numerical Methods: Algorithms and Applications, Prentice-Hall, New Jersey, 2003.
[4] R.G.E. Franks, Modeling and Simulation in Chemical Engineering, John Wiley Interscience, New York, 1972.
[5] C.-E. Fröberg, Introduction to Numerical Analysis, second ed., Addison-Wesley, Reading, 1972.
[6] M. Frontini, E. Sormani, Modified Newton's method with third-order convergence and multiple roots, J. Comput. Appl. Math. 156 (2003) 345–354.
[7] W. Gander, On Halley's iteration method, Am. Math. Month. 92 (1985) 131–134.
[8] M.Ç. Koçak, A class of iterative methods with third order convergence to solve nonlinear equations, J. Comput. Appl. Math., (2007) in press, doi:10.1016/j.cam.2007.02.001.
[9] M.Ç. Koçak, Acceleration of equilibrium calculations via a new technique, in: Presented at 17th International Congress of Chemical and Process Engineering CHISA 2006, 27–31 August 2006, Prague, Czech Republic.
[10] G.A. Nedzhibov, An acceleration of iterative processes for solving nonlinear equations, Appl. Math. Comput. 168 (2005) 320–332.
[11] M. Pakdemirli, H. Boyacı, Generation of root finding algorithms via perturbation theory and some new formulas, Appl. Math. Comput. 184 (2) (2007) 783–788.
[12] J.R. Sharma, A family of third-order methods to solve nonlinear equations, Appl. Math. Comput. 184 (2) (2007) 210–215.
[13] J.F. Traub, Iterative Methods for Solution of Equations, Prentice-Hall, Englewood Cliffs, NJ, 1964.